

Compiling XPath for Streaming Access Policy

Pierre Genevès

`pierre.geneves@inria.fr`

INRIA Rhône-Alpes

(Work done while at IBM T. J. Watson Research Center)

November 2nd, 2005

ACM Symposium on Document Engineering

Bristol, United Kingdom

Outline

Introduction

Example

General Transformations

Conclusion

Introduction

- ▶ More and more data is available using the XML data model
- ▶ XPath is the *very expressive* notation of choice for selection and navigation in XML data
- ▶ Sometimes there is a lot of data!
- ▶ We can still implement *streaming* subsets of XPath efficiently
- ▶ We can even translate all the XPath *reverse axes* into these subsets
- ▶ But the *context position* is not preserved by the translations
- ▶ We translate any XPath expression into another XPath expression without context position references nor reverse axes

Example

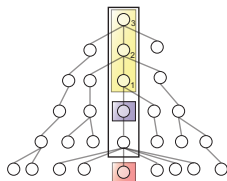
```
/descendant::employee/ancestor::manager[1]
```

Explicitly (in W3C “Core” XPath syntax)

```
distinct-doc-order(  
  let $seq := $root/descendant::employee  
  return  
    for $dot in $seq  
    return  
      let $seq := distinct-doc-order($dot/ancestor::manager)  
      return  
        let $last := count($seq)  
        return  
          for $dot at $rpos in $seq  
          return  
            let $pos := $last - $rpos + 1  
            return  
              if $pos eq 1 then $dot else () )
```

Context position elimination

```
distinct-doc-order(  
  let $seq := $root/descendant::employee  
  return  
    for $dot in $seq  
    return  
      let $seq := $dot/ancestor::manager  
      return  
        let $last := count($seq)  
        return  
          for $dot in $seq  
          return  
            let $rpos := count($dot/ancestor-or-self::manager)  
            return  
              let $pos := $last - $rpos + 1  
              return  
                if $pos eq 1 then $dot else () )
```



Reverse axis elimination

```
distinct-doc-order(  
  let $seq := $root/descendant::employee  
  return  
    for $dot in $seq  
    return  
      let $seq := let $managers := $root/descendant-or-self::manager  
                  return  
                    for $d in $managers  
                    return  
                      if $d/descendant::node() intersect $dot  
                      then $d else ()  
      return  
        let $last := count($seq)  
        return  
          for $dot in $seq  
          return  
            let $rpos := count(let $managers := $root/descendant-or-self::manager  
                               return  
                                 for $d in $managers  
                                 return  
                                   if $d/descendant-or-self::node() intersect $dot  
                                   then $d else ())  
            return  
              let $pos := $last - $rpos + 1  
              return  
                if $pos eq 1 then $dot else () )
```

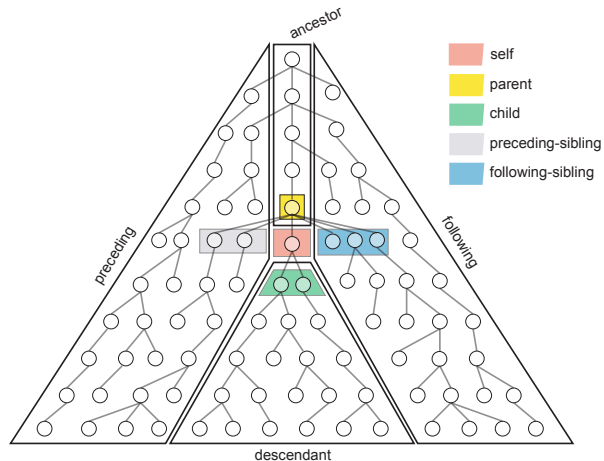
Reverse axis elimination (zoom)

`$dot/ancestor::manager`

became the “search”

```
let $managers := $root/descendant-or-self::manager
return
  for $d in $managers
  return
    if $d/descendant::node() intersect $dot
    then $d else ()
```

Axis partitions



Context position elimination (formally)

$$\begin{aligned}
S[\cdot] &: Expr \rightarrow (Var \rightarrow Expr) \rightarrow Expr \\
S[\text{let } \$Var_{seq} := ForwardAxis :: NodeTest \text{ return } Expr] \rho &= \text{let } \$Var_{seq} := ForwardAxis :: NodeTest \text{ return } \\
&\quad S[Expr] (\rho[Var_{seq} \mapsto ForwardAxis :: NodeTest]) \\
S[\text{let } \$Var_{seq} := ddo(ReverseAxis :: NodeTest) \text{ return } Expr] \rho &= \text{let } \$Var_{seq} := ReverseAxis :: NodeTest \text{ return } \\
&\quad S[Expr] (\rho[Var_{seq} \mapsto ReverseAxis :: NodeTest]) \\
S[\text{let } \$Var_{seq} := ddo(Expr_1) \text{ return } Expr_2] \rho &= \text{let } \$Var_{seq} := ddo(S[Expr_1] \rho) \text{ return } \\
&\quad S[Expr_2] (\rho[Var_{seq} \mapsto S[Expr_1] \rho]) \\
S[\text{for } \$Var_{dot} \text{ at } \$Var_{pos} \text{ in } \$Var_{seq} \text{ return } Expr] \rho &= \text{for } \$Var_{dot} \text{ in } \$Var_{seq} \text{ return } \\
&\quad \text{let } \$Var_{pos} := Expr_{pos} \text{ return } S[Expr] \rho
\end{aligned}$$

where $Expr_{pos}$ is given by the following table:

$\rho(Var_{seq})$	$Expr_{pos}$
<code>self :: NodeTest</code>	1
<code>child :: NodeTest</code>	count (preceding-sibling :: NodeTest) + 1
<code>parent :: NodeTest</code>	1
<code>ancestor :: NodeTest</code>	count (ancestor-or-self :: NodeTest)
<code>ancestor-or-self :: NodeTest</code>	count (ancestor-or-self :: NodeTest)
other Expr	let $\$Var_d := \Var_{dot} return count (for $\$Var_{dot}$ in $\$Var_{seq}$ return if node-before($\$Var_{dot}, \Var_d) then $\$Var_{dot}$ else ()) + 1 where $\$Var_d$ is a fresh variable

Reverse axis elimination (formally)

$F[\cdot]: Expr \rightarrow Expr$

$F[\$Var / ReverseAxis::NodeTest]$

= let $\$Var_s := \$root / descendant-or-self::NodeTest$ return
 for $\$Var_d$ in $\$Var_s$ return
 if $\$Var_d / ForwardAxis::node()$ intersect $\$Var$ then $\$Var_d$ else ()
with Var_s, Var_d "fresh", and $ReverseAxis$ determined by

<i>ReverseAxis</i>	<i>ForwardAxis</i>
parent	child
ancestor	descendant
ancestor-or-self	descendant-or-self
preceding-sibling	following-sibling
preceding	following

Conclusion

Limitations:

- ▶ *Outermost* context position should still be accessed in the normal way.
- ▶ Depends crucially on XPath *version 1.0* restrictions on path steps.
- ▶ Search should *not* be implemented naïvely; in general optimizations are still needed.

Outcome:

- ▶ XPath (1.0) can be fully supported even for streaming!

Appendix

Questions?

W3C XPath/XQuery Formal Semantics

VarRefpos is bound to the position of the item in the input sequence:

$\text{dynEnv} \vdash \text{Expr}_1 \Rightarrow \text{Item}_1, \dots, \text{Item}_n$

$\text{statEnv} \vdash \text{VarRef}$ of var expands to Variable

$\text{statEnv} \vdash \text{VarRefpos}$ of var expands to Variablepos

$\text{dynEnv} + \text{varValue}(\text{Variable} \Rightarrow \text{Item}_1; \text{Variablepos} \Rightarrow 1) \vdash \text{Expr}_2 \Rightarrow \text{Value}_1$

...

$\text{dynEnv} + \text{varValue}(\text{Variable} \Rightarrow \text{Item}_n; \text{Variablepos} \Rightarrow n) \vdash \text{Expr}_2 \Rightarrow \text{Value}_n$

$\text{dynEnv} \vdash \text{for VarRef at VarRefpos in Expr}_1 \text{ return Expr}_2 \Rightarrow \text{Value}_1, \dots, \text{Value}_n$